Ain Shams University
Faculty of Computer & Information Sciences
Computer Science Department

Developing an Approach for Solving Ambiguity in Requirements Specification to UML Conversion

A Thesis

Submitted to Computer Science Department, Faculty of Computer & Information Sciences Ain Shams University, Cairo, Egypt in Partial Fulfillment of the Requirements for Master Degree in Computer Science

By

Somaia Osama Mohamed Rashad

Teaching Assistant, Computer Science Department, Akhbar El-Yom Academy

Supervised By

Prof. Dr. Mostafa M. Aref
Professor of Computer Science
Faculty of Computer and Information Sciences, Ain Shams University

Dr. Safia Abbas
Associate Professor of Computer Science
Faculty of Computer and Information Sciences, Ain Shams University

Cairo – 2018

ABSTRACT

Requirements analysis phase is the first step of software building process. This phase made a Software Requirements Specification document (SRS). SRS document will be the base for development team to build a software application. Requirement Specification is a document that acts as a medium between system developer and users. The document contains sentences and statement that state the functional and nonfunctional requirements. The success of the software is mostly dependent on how well the users' requirements have been understood and converted into suitable functionalities in the software [1].

Using UML models, information can be effectively represented. These models are useful for realizing the problems, interacting with stockholders and making documentation. Also, in all phases of software development process these models can be used effortlessly. Converting natural language to UML models automatically save time, effort and cost. But, Ambiguity is a critical issue in the software requirement specifications. Ambiguity is best defined by 'having more than one meaning', and is inherent to natural language. When we communicate, ambiguity may lead to noise in the communication channel [2]. Through the noise, the message of the sender may not be received as intended by the receiver. Humans recognize and resolve ambiguous words and descriptions by accessing all possible meanings that are stored in the brain. Ambiguity occurs when different readers can interpret a sentence differently. Usually, the users express their requirements in natural language statements that initially appear easy to represent. However, being represented in natural language, the statement of requirements often tends to suffer from ambiguities. Stakeholders are frequently not even conscious that there is an ambiguity in a requirement. Each stakeholder understands from reading the requirements an explanation that diverges from that of others, without knowing this divergence. So, the software developers design and develop a system that does not work as proposed by the users, but the developers justly think they have done the requirements. Because of the inherent ambiguity of NL, it is often difficult to prove properties on NL requirements and it causes high error rate in converting to UML [3]. Therefore, it is necessary for a tool that can detect and remove ambiguity of SRS document.

The explanation of the system functions should to be unambiguous, implying that it is free of dissimilar interpretations. If the explanation has more than one possible interpretation, the software programmer may understand an ambiguous word in a manner the customer did not mean. This can lead to a system that does not meet the requirements of the customer. Dissimilar interpretations frequently remain undiscovered till later phases of the software life cycle, when design and implementation selections shape the specific understandings. It costs 50-200 times as much to rectify an error late in a system project compared to when it was presented [4]. Most previous work on ambiguity in requirements engineering tried to address the problem from providing users with a restricted natural language, tool, or handbook to assist with writing less ambiguously. Therefore, it is necessary for an approach that can detect and resolve the ambiguity of SRS document. It can help system analyst in determining and removing the ambiguity of a requirement statement.

In this thesis, we illustrate an automated approach for detecting and resolving ambiguities that cause a high risk of misunderstanding by several readers and lead to confusion, waste of both effort and time and rework. Sentences in a natural language requirements specification document that have ambiguity are initial detected automatically from the SRS document and ambiguity type is determined. Sentences that include ambiguity are then resolved automatically also by

resolving algorithm based on a set of rules that we collected from training data. We implemented a tool for Detecting and Resolving Ambiguity (DARA), in order to clarify and estimate our approach. The tool focuses on Lexical, Referential, Coordination, Scope and Vague ambiguity, it can classify between them and it can calculate the percentage of each ambiguity type. DARA does not oblige the requirement engineers to write using a particular standard or style. Due to the training data, it turned out that we are currently able to develop a system which can recognize and resolve ambiguity in requirements specifications. We determine on the results of a collection of requirement specification documents to evaluate the performance and utility of the approach. It transpired that increasing the size of our training data, would improve the accuracy.

ACKNOWLEDGMENTS

The greatest thanks to Allah before and after. Firstly, I would like to express my sincere gratitude to my supervisor Prof. Dr. Mostafa M. Aref for the continuous support of my study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my master study. Besides my supervisor, I would like to thank the rest of my thesis committee: Prof. Dr. Zaki Taha, and Prof. Dr. Amr Badr for their insightful comments and encouragement, but also for the questions which incented me to widen my research from various perspectives. A special thanks to my family. Words cannot express how grateful I am to my parents and my brothers, Muhammad and Ebrahim and my sister, Sarah for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis and my life in general. Last but not the least, I would like to thank my friends and my students for providing me with unending inspiration and motivation. This accomplishment would not have been possible without All of them. Thank you.

PUBLICATIONS

Parts of this thesis are published as original papers in the following references:

1. Somaia Osama, Safia Abbas, Mostafa Aref "Solving Ambiguity in Requirements Specification to UML Conversion", ESOLEC'14, The Fourteenth Conference on Language, the Egyptian Society of Language Engineering, Faculty of Engineering, Ain Shams University Engineering, December 3-4, 2014, Cairo, Egypt
2. Somaia Osama, Safia Abbas, Mostafa Aref "Developing an Approach for Solving Ambiguity in Requirements Specification to UML Conversion", ESOLEC'15, The Fifteenth Conference on Language, the Egyptian Society of Language Engineering, Faculty of Engineering, Ain Shams University Engineering, December 9-10, 2015, Cairo, Egypt
3. Somaia Osama, Safia Abbas, Mostafa Aref "Ambiguity Detection and Resolving in Natural Language Requirements", ESOLEC'16, The Sixteenth Conference on Language, the Egyptian Society of Language Engineering, Faculty of Engineering, Ain Shams University Engineering, December 7-8, 2016, Cairo, Egypt
4. Somaia Osama, Mostafa Aref "Detecting and Resolving Ambiguity Approach in Requirement specification: Implementation, Results and Evaluation", International Journal of Intelligent Computing and Information Science 2018, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt, Accepted for publication.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

LIST OF ABRIVIATIONS

Abbreviation

NLP Natural Language Processing

NLRS Natural Language Requirements Speciation

NLTK Discourse Relations

OCL Object Constraint Language

OOD Object Oriented Design

SBVR Semantics of Business Vocabulary and Rules

SRS Software Requirements Specification

WSD Word Sense Disambiguation

XML eXtensible Markup Language

1

Introduction

CHAPTER OUTLINE

- **Overview**
- **Problem Definition**
- **Aims**
- **Contributions**
- **Structure of the thesis**

Chapter 1

Introduction

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view. The process to gather the software requirements from client, analyze and document them is known as requirement engineering. The goal of requirement engineering is to develop and maintain sophisticated and descriptive 'System Requirements Specification' document. Requirement Engineering Process It is a four step process, which includes Feasibility Study, Requirement Gathering, Software Requirement Specification and Software Requirement Validation [4].

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven project, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules [5]. Used appropriately, software requirements specifications can help prevent software project failure [6]. The software requirements specification document enlists enough and necessary requirements that are required for the project development [7]. To derive the requirements, the developer needs to have clear and thorough understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase. It describes the required behavior of a software product, and is often specified as a set of necessary requirements for project development. A software requirements specification should Correct, Unambiguous, Complete, Consistent, Ranked for importance and/or stability, Verifiable, Modifiable and Traceable. So an ideal SRS should clearly state the requirements without introducing any ambiguities. Unfortunately, it is impossible to avoid the ambiguous SRSs since they are often described using natural languages. A requirement is ambiguous if it can be interpreted in multiple ways. Ambiguous requirements can be a major problem in software development [8]. Project participants tend to subconsciously disambiguate requirements based on their own understanding without realizing that they are ambiguous. As a result, different interpretations often remain undiscovered until later stages of the software life cycle, when design and implementation choices materialize the specific interpretations.

Typically, system requirements are written in a natural language. Even if system requirements are described in a formal language, the first copy is always described in a natural language. A natural language software requirements specification improves the communication among all the stakeholders. When software requirements written in natural language, this leading them into ambiguity, inconsistency and incompleteness. Ambiguity problem can be identified at an early phase of the development process, therefore decreasing the development costs and the number of resulting errors. Although requirements written in natural language have the benefit of being clear to both customers and developers, they can naturally as well be incomplete, vague, and ambiguous.